9th International Symposium on Display Holography (ISDH 2012) IOP Publishing
Journal of Physics: Conference Series **415** (2013) 012042 doi:10.1088/1742-6596/415/1/012042

Challenges in using GPUs for the real-time reconstruction of digital hologram images

P R Hobson, J J Nebrensky and I D Reid

Centre for Sensors and Instrumentation, School of Engineering and Design, Brunel University, Uxbridge UB8 3PH, UK

E-mail: peter.hobson@brunel.ac.uk

Abstract. In-line holography has recently made the transition from silver-halide based recording media, with laser reconstruction, to recording with large-area pixel detectors and computer-based reconstruction. This form of holographic imaging is an established technique for the study of fine particulates, such as cloud or fuel droplets, marine plankton and alluvial sediments, and enables a true 3D object field to be recorded at high resolution over a considerable depth.

The move to digital holography promises rapid, if not instantaneous, feedback as it avoids the need for the time-consuming chemical development of plates or film film and a dedicated replay system, but with the growing use of video-rate holographic recording, and the desire to reconstruct fully every frame, the computational challenge becomes considerable. To replay a digital hologram a 2D FFT must be calculated for every depth slice desired in the replayed image volume. A typical hologram of ~100 μm particles over a depth of a few hundred millimetres will require $O(10^3)$ 2D FFT operations to be performed on a hologram of typically a few million pixels.

In this paper we discuss the technical challenges in converting our existing reconstruction code to make efficient use of NVIDIA CUDA-based GPU cards and show how near real-time video slice reconstruction can be obtained with holograms as large as 4096 by 4096 pixels. Our performance to date for a number of different NVIDIA GPU running under both Linux and Microsoft Windows is presented. The recent availability of GPU on portable computers is discussed and a new code for interactive replay of digital holograms is presented.

1. Introduction

In the last decade digital in-line holography has replaced silver-halide based techniques for the 3D recording of volumes of small particles. It has applications in a number of areas of science and engineering, for example cloud or spray droplets [1], marine plankton [2] and sediments [3]. Digital holography replaces the older technique, which primarily used high-resolution silver-halide emulsions, with direct recording onto a solid-state pixel sensor such as a CCD or CMOS array. There are a number of different mathematical techniques that can be used to calculate the replayed image volume; ours is based upon the convolution approach [4]. To replay the image volume encoded by a digital hologram requires that the two-dimensional discrete Fourier transform of the hologram is multiplied

by a phase factor related to the recording beam and the distance of the replaced depth slice and then the image is created by taking the inverse discrete Fourier Transform and calculating its magnitude. A typical CCD or CMOS sensor used in digital holography has a few million pixels, although individual sensors with over eighty million pixels have been made. To fully reconstruct the volume recorded on a digital hologram requires 2D depth-slices at sub-millimeter intervals to be reconstructed over the entire recorded depth which is often a few hundred millimeters in extent. Thus for a single hologram of order one thousand individual 2D images will be replayed. The computational task, even with modern multi-core processors and efficient discrete Fourier transform algorithms such as FFTW [5] is considerable. In previous work [7], [8] we have exploited the trivially parallel nature of the hologram replay to enable us to use computational grid resources. Unfortunately the stochastic nature of submission to grid resources, via a broker, and the possibility of being queued until the validity of a job's X.509 proxy certificate expires means that the initial significant speed advantage over a single local computational node is rarely maintained and quite often not all of a volume is reconstructed. In earlier work a number of different jobs each reconstructing ten, fifty or one hundred sequential depth slices, for a total of 2200 planes for each run, were submitted to nodes on the EGEE grid [9]. An initial large advantage over a single processor was seen but the rate dramatically slowed in all cases and only a fraction of all the jobs completed, behaviour which is typical of current scientific production grids.

With the advent in recent years of powerful and increasingly inexpensive GPU chips considerable effort from ourselves [10] and many other researchers (see for example [11], [12], [13]) has been expended in porting, or developing, digital hologram replay code to run efficiently on these systems. In this paper we describe our latest work using NVIDIA Tesla and Fermi class GPU systems, the optimization techniques that we have deployed to achieve very significant replay speed, and the prospects for further improvement. Finally we speculate on whether a hybrid approach of GPU and distributed (grid or cloud) based computing may be optimum when the task of both replay and object extraction is considered.

2. Recording and replaying a digital hologram

We consider in this paper only the recording and replay of in-line (Fraunhofer) holograms recorded digitally on a pixel area sensor. Other forms of digital hologram, for example those recorded in off-axis holography, do not differ in the essential principles of replay.

2.1. Hologram recording

We recorded in-line holograms of objects in water using an 8 megapixel camera (ATMEL Camelia 8M, 2300 by 3500 10 μm-square pixels with 12-bit depth) with a collimated beam from a c.w. HeNe laser (λ=633 nm, 1 mW). As in earlier work [10] we recorded a sample volume consisting of cenospheres mostly of 100-300 μm dia. (Fillite Trelleborg Specialty Grade (High Alumina) SGHA 500) dispersed in a water tank (figure 1). A variety of other subjects, for example air bubbles in water were also used as test objects.

More recently, dynamic objects, such as digitalis seeds set in motion by breath [14], were recorded using a fast pulsed green laser (figure 2). The laser is a custom doubled Nd-YAG from Elforlight Ltd, model number LP3-10-532. This produced single longitudinal-mode pulses with a wavelength of 532 nm, duration of 2 ns and an adjustable energy of a few millijoules. The beam was then attenuated using a Newport 935-10 high-power variable attenuator and reflected, using a BK7 wedge, towards a beam expander telescope. The expanded beam then traversed a sample tank with BK7 flat windows and the resulting hologram was recorded using the same ATMEL camera.

2.2. Hologram replay

Our general approach has been summarised elsewhere [5]. The process is computationally heavy as it requires multiple 2D complex FFT to be calculated. Regarding only the reconstruction of a digital hologram, and ignoring the challenging and similarly computationally intensive task of locating "regions of interest" (ROI) within the replayed volume, it can be seen that every depth slice may be calculated in parallel. Since the initial Fourier transformed hologram is common to any depth slice calculation there are clearly potential advantages in doing this step once and making the transformed data available to all processors. With the recent availability of very powerful GPU processors and the porting of the FFTW algorithm to the NVIDIA CUDA library, most recent work in the digital holography field has made use of such GPU based systems. In the next section we describe results of our work on porting our existing replay code *HoloReco* [15] to the NVIDA CUDA environment.

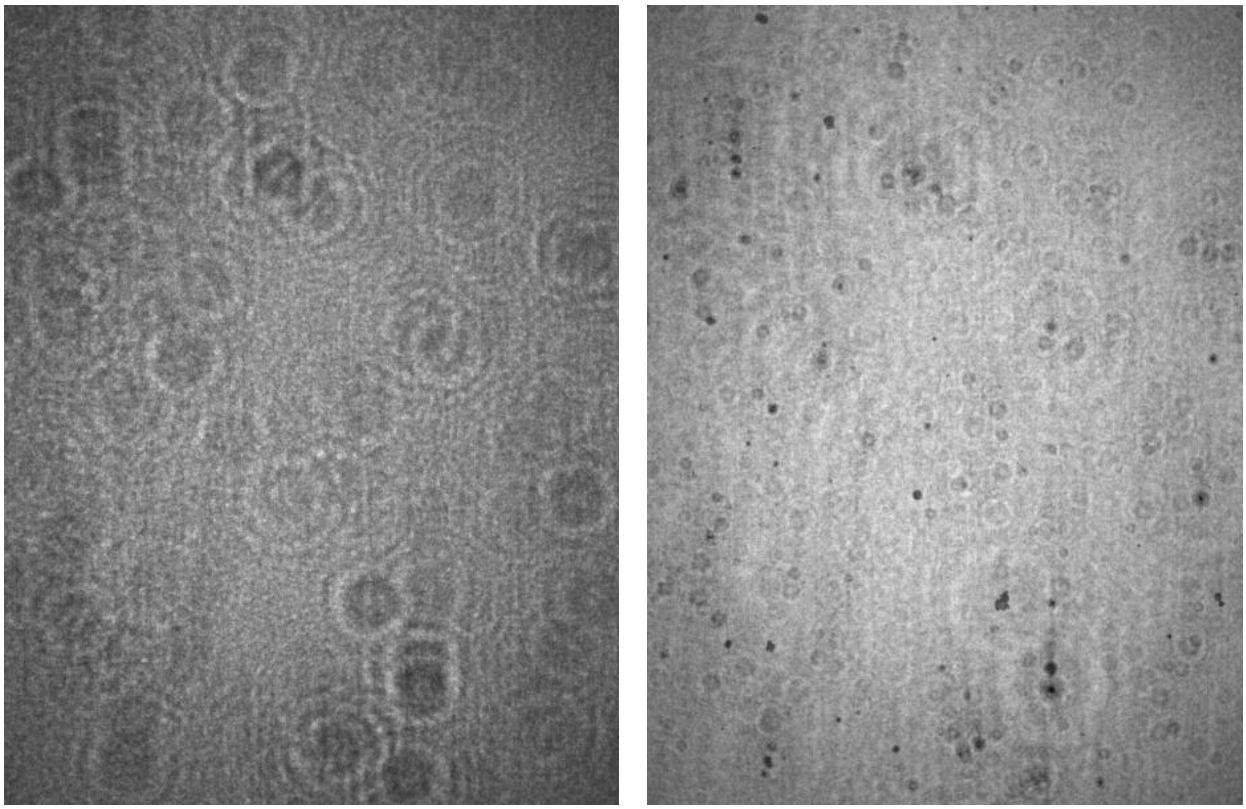

Figure 1. (Left): An extract (10 mm wide) from a digital hologram of cenospheres in water. (Right): The region of a numerically-reconstructed arbitrary plane through the sample volume. The smallest particles imaged here are approximately 30 μm in size [9].

3. Porting to the NVIDIA CUDA environment.

The original *HoloReco* code carried out reconstruction separately for every depth slice. Performing the FFT of the input hologram and maintaining the result to be used subsequently for a series of different depth slices eliminated all but one of the time intensive serial transfers from host to GPU memory.

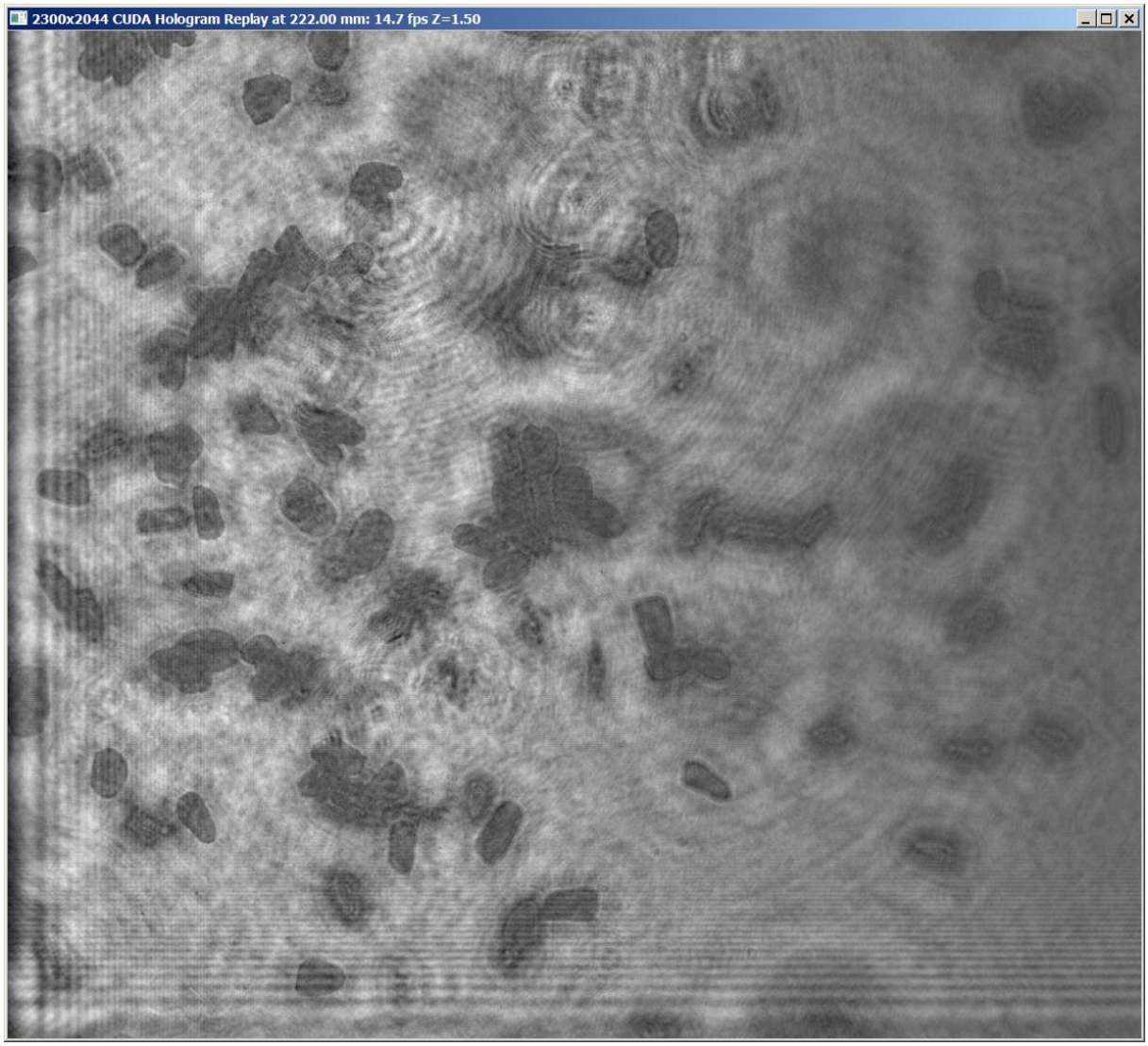

Figure 2. Window capture of the replay of a pulsed-laser hologram on a laptop computer equipped with a GT 555M GPU. The image shows digitalis seeds (~1 mm long) scattered by breath. The original hologram was cropped to 2300x2044 pixels and the image has been zoomed by a factor of 1.5 in the 1187x1080 window.

Another significant improvement to the processing speed was finding a way to parallelise the transfer function (unique to each depth plane). This allowed all computations to be carried out in the GPU and thus no bulk data movement is needed until the output stage

3.1. Tesla-based system with no video display

Using a system containing three NVIDIA Tesla S1070 (240 thread processors per GPU core, shaders clocked at 1.30 GHz) we were able to process our hologram to files at 13.6 planes/second. This should be compared to our original code running on an eight-core Xeon E5420 (2.50 GHz) based system where a speed of only 0.057 planes/second was achieved. Thus code refactoring and using three GPUs

provided a factor of nearly 240 improvement in reconstruction speed. These early results used V2.2 of the CUDA libraries (including FFTW).

3.2. Fermi-based systems with video display

There are two clearly different user cases for hologram data visualisation:

- An expert user (e.g. a Marine Biologist) searches through a video stream of replayed hologram depth slices looking for "interesting" objects
- An image-processing system searches through replayed hologram image slices looking for "objects". The set of detected objects is then sent to a classifier.

A better use of the human expert is for them to locate quickly holograms that contain some objects of interest, for example a zooplankton species, and then to create a selected list of holograms to be systematically replayed in full and processed by an image-processing system and object classifier. Alternatively the expert user could look, in context, at those regions of the replayed volume in which an automated system had previously located and classified objects. Some Tesla and many Fermi-class GPUs have a video output and an OpenGL compatibility library is included in the CUDA development environment. For human visualisation this is ideal since one can efficiently copy the reconstructed hologram to the video buffer and overall performance is not compromised. As well, the OpenGL library allows simple panning and zooming of the image within the display window. The sequence of events for each replayed frame is shown in figure 3.

Table 1 lists the video rates we achieved for three different holograms (two cropped from the original) on four separate GPUs with integrated video output. Notable is the performance of the GT 555M GPU, which is designed for portable systems and in our case is part of a Dell XPS17 laptop computer which consumes just 80 W of power whilst continuously calculating and displaying the reconstructed images.

In practice, the replay software (*HoloMovie*) has a simple interface. Control and data parameters, such as laser wavelength, pixel spacing, input file-name, and reconstruction step size and depth range are read, as in *HoloReco*, from a simple text file. The programme calculates the window size which will fit the largest reconstructed image on the screen and begins the cycle of reconstruction and passing the image data to the display system. User controls are limited to manually changing the reconstruction depth or choosing to loop continuously back and forward in the range set by the parameter file, zooming the image in and out within the window, and panning the window across the image. This allows the operator to scan the reconstruction quickly and to examine closely any interesting features which are discovered.

Table 1. Reconstruction and replay rates achieved with several modern GPUs for three separate histogram sizes (zero-padded sizes in parentheses). All models are Fermi-class devices and all operating systems are 64-bit..

GPU Model	Thread Processors	Operating System	CUDA Version	Driver Version	2048x1389 (2048x2048)	2300x2044 (4096x2048)	2300x3500 (4096x4096)
GT 440	96	Windows 7	4.1	301.42	24.2 fps	13.6 fps	7.4 fps
GT 555M	144	Windows 7	4.1	301.42	25.1 fps	14.7 fps	9.3 fps
GTX 460	336	Scientific Linux 5	4.1	302.07	30.0 fps	20.0 fps	12.0 fps
C 2070	448	Ubuntu 11.10	4.2	295.53	39.8 fps	29.2 fps	15.9 fps

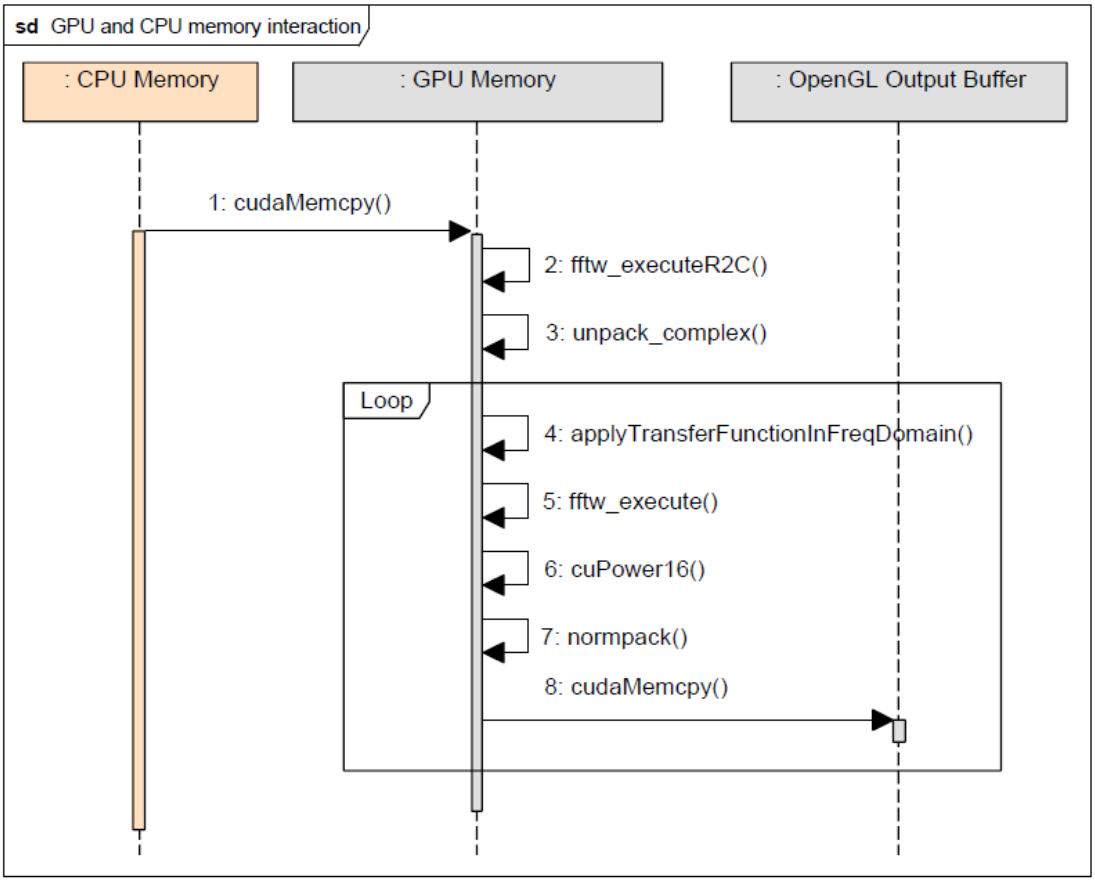

Figure 3. A UML representation of the sequence of key events in our software that occur during the processing of a digital hologram. If a number of frames from the same hologram are replayed to video then the sequence numbers 4 through 8 are repeated as often as required.

3.3. Profiling the code

The CUDA SDK also includes a profiler for mapping GPU usage which is a powerful tool for locating and understanding which operations dominate the GPU processor time. We ran the CLI version to get a timestamp snapshot of the programme running for 15 seconds on a 4-Core Xeon computer running Ubuntu 11.04 and equipped with a Fermi-based C2070 GPU. This gave the start-up and 50 iterations of the display loop (at ~4.4 frames/s; without the profiler it ran at ~16 frames/s). The hologram was 2300x3500 and zero-padded to 4096x4096 for processing. Table 2 shows information on GPU usage for this run.

The timing data show that, once the initial file is read, the majority of the GPU time is spent manipulating data – performing the transfer function to prepare for the reverse FFT and converting back from the complex transform to real magnitudes.

Table 2. (a) GPU and CPU procedure call timings and GPU occupancy for the initialisation step common to all slices. (b) GPU and CPU timings and GPU occupancy, averaged over 50 iterations, of the procedure calls that are executed for each depth slice calculated and then displayed. Note that there are two occupancy figures for the FFT because there are two separate kernels executed for each FFT; one executes once, the other twice.

(a) GPU kernel	Number of calls	GPU time (μs)	CPU time (μs)	GPU occupancy	Comment
memset32_post	3	6.7	27	0.042	
memcpyHtoD	1	12203	12313		Copy data from Host to Device
Forward FFT operations	3	5011	36	0.667/0.417	
Post-process	1	2379	5	1.000	
memcpyHtoD	1	2.0	7		Parameters for unpacking
copy_kernel	4095	11123	18524	1.000	cudaBlas unpacking FFT
insert_conj	1	15609	4	0.167	Reconstruct 2nd half of FFT
setUpX22_kernel	1	421.1	9	0.021	Calculate a lookup table for transfer

(b) GPU kernel	Number of calls/loop	Average GPU time/loop (μs)	Average CPU time/loop (μs)	Av. GPU occupancy	Comment
doInnerLoop	4096	20780	19891	0.167	Parallelized Transfer Function
Reverse FFT operations	3	9898.9	26.0	0.667/0.333	
cuPower16_kernel	1	22333	4.6	0.500	Extract amplitudes from complex & remove zero-padding
memcpyDtoH	1	2.6	40.1		Return subroutine results
memcpyDtoH	1	2.1	33		Return subroutine results
memcpyHtoD	1	1.6	5.4		Parameters for minmax to GPU
thrust::minmax_element	2	516.0	11.7	0.500	Find min & max
memcpyDtoH	1	2.0	539.0		Return subroutine results
setminmax	1	4.3	6.5	0.021	Place scaling factors in device
packfloat	1	5128.4	4.4	0.167	Create scaled, packed pixels
memcopyDtoD	1	272.4	7.8		Copy to OpenGL buffer

4. Discussion

With our code achieving a replay frame rate approaching standard video for holograms as large as 4096x4096 pixels and significantly exceeding it for 2048x2048 pixel holograms, there is no doubt that for visualisation a GPU is far superior to a conventional CPU or a computational grid. Recent advances (for example the GTX460 and GTX560 based systems) in inexpensive "desk-top Fermi" processor GPU systems, developed primarily for the high-end computer gaming community, now enable a very impressive price-to-performance ratio to be obtained for a modest monetary outlay. Even more significant are the most recent Kepler-class systems exemplified by the GTX 680 with 1536 thread processors and power consumption less than 200 W. A recently released laptop computer (Samsung Chronos 17) improves on our GT 555M system with a Kepler based GT 650M GPU, containing 384 processors for twice the performance at the same power consumption. Clearly it is

now becoming feasible to take considerable processing power into the field rather than needing to bring data back to a compute farm.

Even further improvements lie on the horizon, with the upcoming CUDA 5 technology and the second-generation Kepler GPUs providing new techniques such as dynamic parallelism, bringing the need to master these concepts for maximum performance. However the picture is less clear when considering the overall task of replaying a digital hologram and then searching in 3D for in-focus images of "interesting" objects, extracting these and then classifying them.

Much existing code, including extensive image processing and classification libraries, would need to be ported to a GPU to generate a significant speed increase. Nonetheless the challenge lies not in the porting but in the code refactoring needed to make efficient use of the GPU. In our earlier work [10] the naive approach of recompiling our unmodified *HoloReco* code with the CUDA FFTW libraries resulted in only a factor of six improvement in processing speed and only extensive code refactoring coupled with detailed profiling has enabled the current factor of 240 to be achieved. In addition the task of locating objects in a replayed 3D volume is not a trivially parallel one. Several correlated depth slices need to be considered in order to locate, using some appropriate focusing metric [16], the best one so that each object may be extracted for subsequent classification. We hypothesise that a hybrid approach, with replay using a GPU and subsequent processing of the datasets using grid or commodity cloud computing may be the most appropriate combination of the two different architectures.

References

[1] Yang Yan, Kang Bo-seon 2011 Opt.Lasers Eng 49 1254–63
[2] Sun H, Benzie P W, Burns N, Hendry D C, Player M A and Watson J 2008 Phil. Trans. R. Soc. A 366 1789–806
[3] Graham G W, Smith W and Nimmo A M 2010 Limnol.Oceanogr.Meth. 82 1–15
[4] Kreis T M, Adams M and Juptner W P O 1997 Proc. of SPIE 3098 224–33
[5] Reid I D, Nebrensky J and Hobson P R (2012) Challenges in using GPUs for the reconstruction of digital hologram images *Advanced Computing and Analysis Techniques in Physics Research* September 5-9, 2011, Uxbridge, London to appear in *Journal of Physics, Conference Series*
[6] Frigo M and Johnson S G 2005 Proc. IEEE 93 216–31
[7] Nebrensky J J, Hobson P R and Fryer P C 2005 Proc. of SPIE 5775 285–96
[8] Nebrensky J J and Hobson P R 2006 Proc. of SPIE. 6252 62521I.1–6
[9] Nebrensky J J and Hobson P R 2009 Replay of Digitally-Recorded Holograms Using a Computational Grid [available] http://bura.brunel.ac.uk/handle/2438/3443
[10] Nebrensky J J, Hobson P R and Reid I D 2009 Digitally-Recorded Hologram Replay - a Comparison Between a Computational Grid and a GPU Based System, EOS Topical Meeting on Blue Photonics, Aberdeen, UK 18-19 August 2009
[11] Pandey N, Kelly D P, Naughton T J and Hennelly B M, 2009 Proc. of SPIE 7442 744205–1
[12] Zulfiqar A, Hyun-Eui Kim, Dongbiao Han, Jae-Hyeung Park and Nam Kim 2011 3D Res. 2 1–5
[13] Zhuqing Zhu, Min Sun, Heping Ding, Shaotong Feng and Shouping Nie 2009 Fast numerical reconstruction of digital holography based on graphic processing unit [Available] http://dx.doi.org/10.1109/CLEOPR.2009.5292249
[14] Hobson P R, Reid ID and Wilton JB 2012 Visualising breath using digital holograms, this conference
[15] [Available] http://holoreco.sourceforge.net/
[16] Meinecke T, Sabitov N and Sinzinger S 2010 Appl.Opt. 49 2446–55